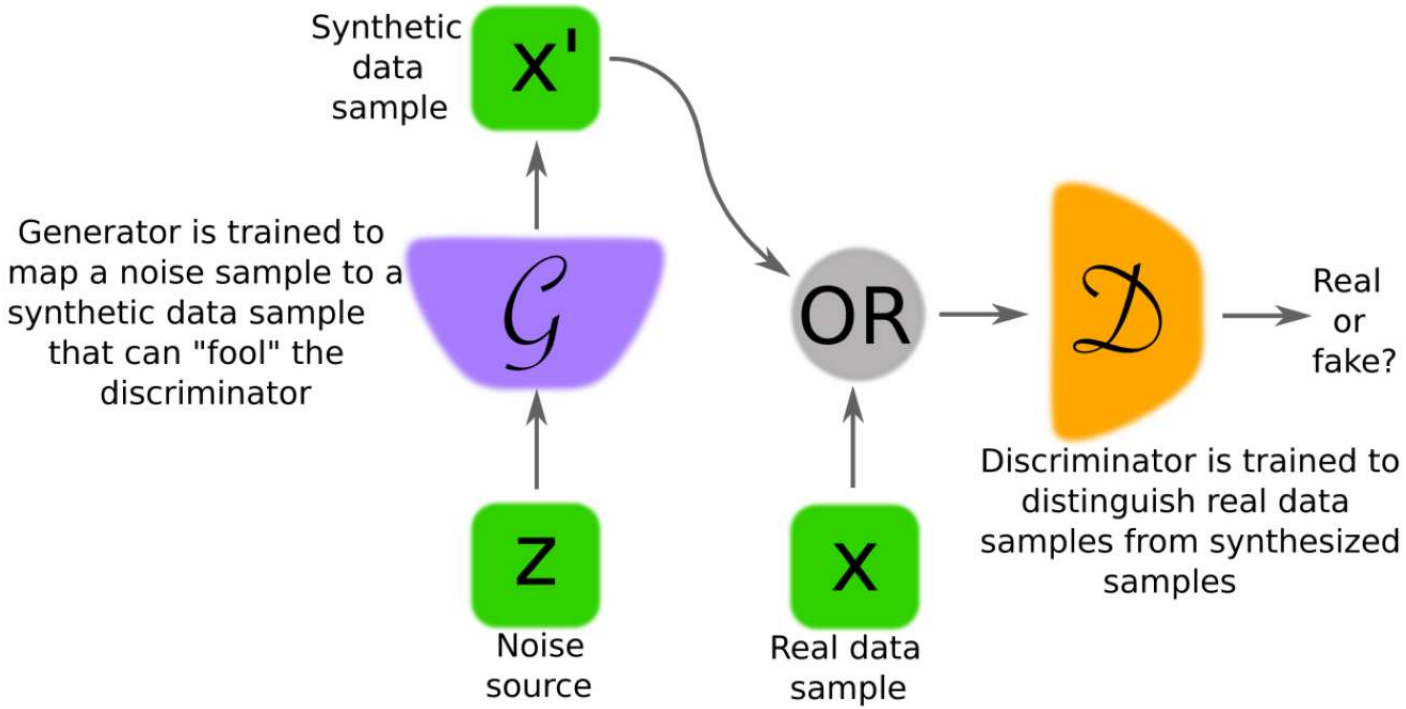
A Brief Summary of GANs

https://github.com/ijleesw

GAN의 기본 컨셉

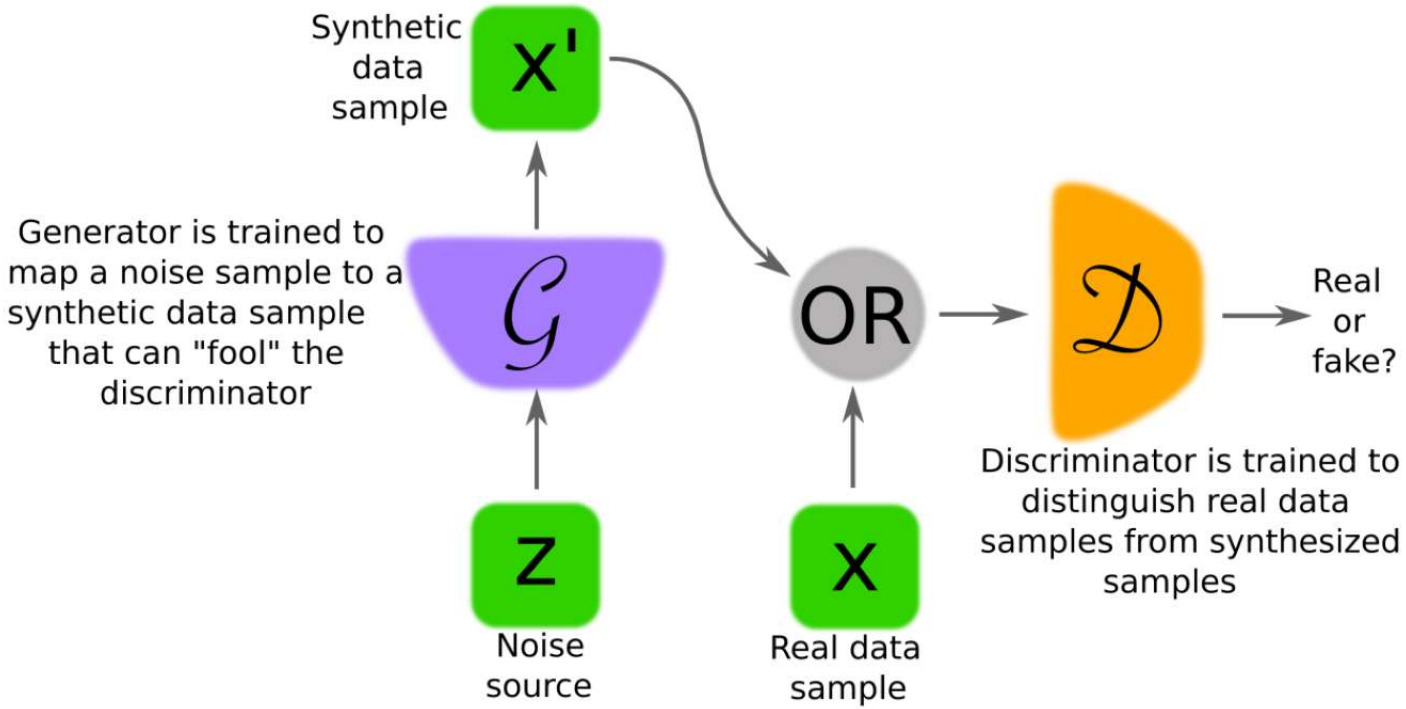


- noise sample에서 image를 생성하는 Generator(G),
  real image와 fake image를 구분하는 Discriminator(D)로 구성.

- Loss function : $\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(x)))]$

  (Repeat : V를 최대화하도록 D를 업데이트한 뒤
  다시 V를 최소화하도록 G를 업데이트)

GAN의 기본 컨셉



- 장점 : 다양한 distribution을 습득 가능.
  (현실세계 데이터의 density estimation을 근사하는 게 GAN의 핵심!!)

- 단점 : explicit density estimation은 구할 수 없음.
  G의 training 정도에 대한 적절한 measure 존재 x.

DCGAN

(noise sample z에서 fake image G(z)가 생성되는 과정)

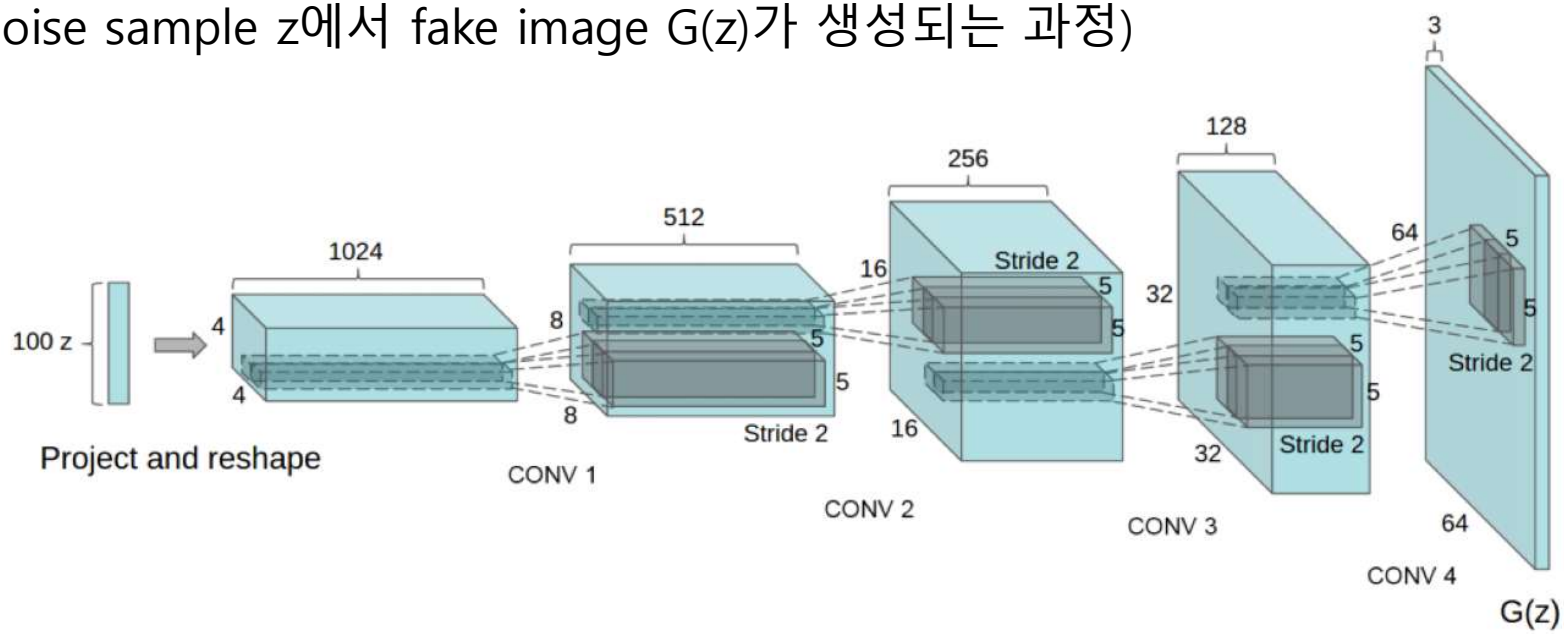


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

DCGAN

- GAN은 Generator와 Discriminator에서 MLP를 사용. But, DCGAN은 CNN을 사용. 몇 가지 테크니컬한 변경점도 있음. (위 링크 참고)

- 현재 거의 모든 GAN의 뼈대가 됨.

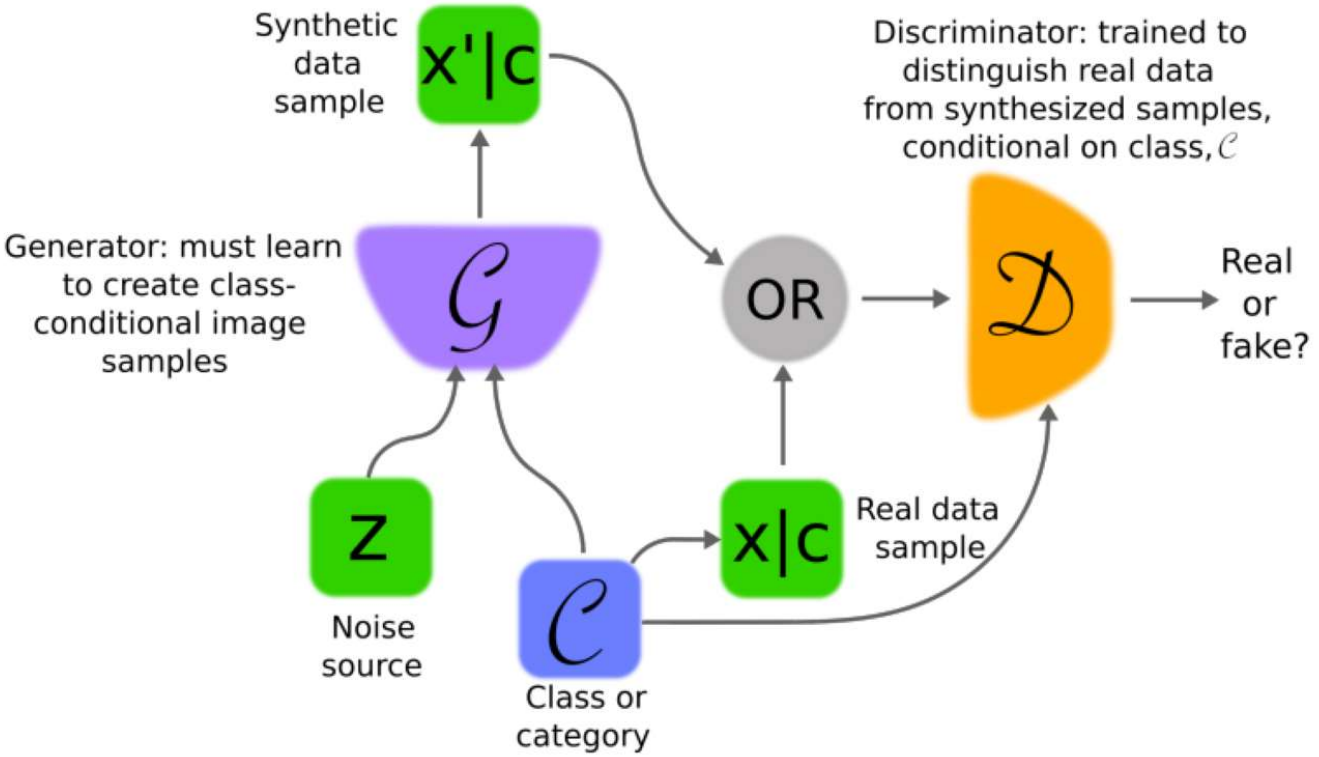
- 많은 데이터, 고화질 데이터에도 적용 가능.

- Discriminator를 SVM과 함께 classification에 사용 가능.
  K-means와 비슷한 accuracy를 보인다.

- noise sample vector의 값을 조금씩 계속 바꿔줄 때 이미지가 부드럽게 변함
  => image memorization이 일어나지 않는다는 증거!

- noise sample에 대해 vector arithmetic 가능.
  ([smiling woman이 나오는 noise vector - neutral woman이 나오는 n.v.
  + neutral man이 나오는 n.v.]를 G에 넣으면 smiling man이 나온다.)

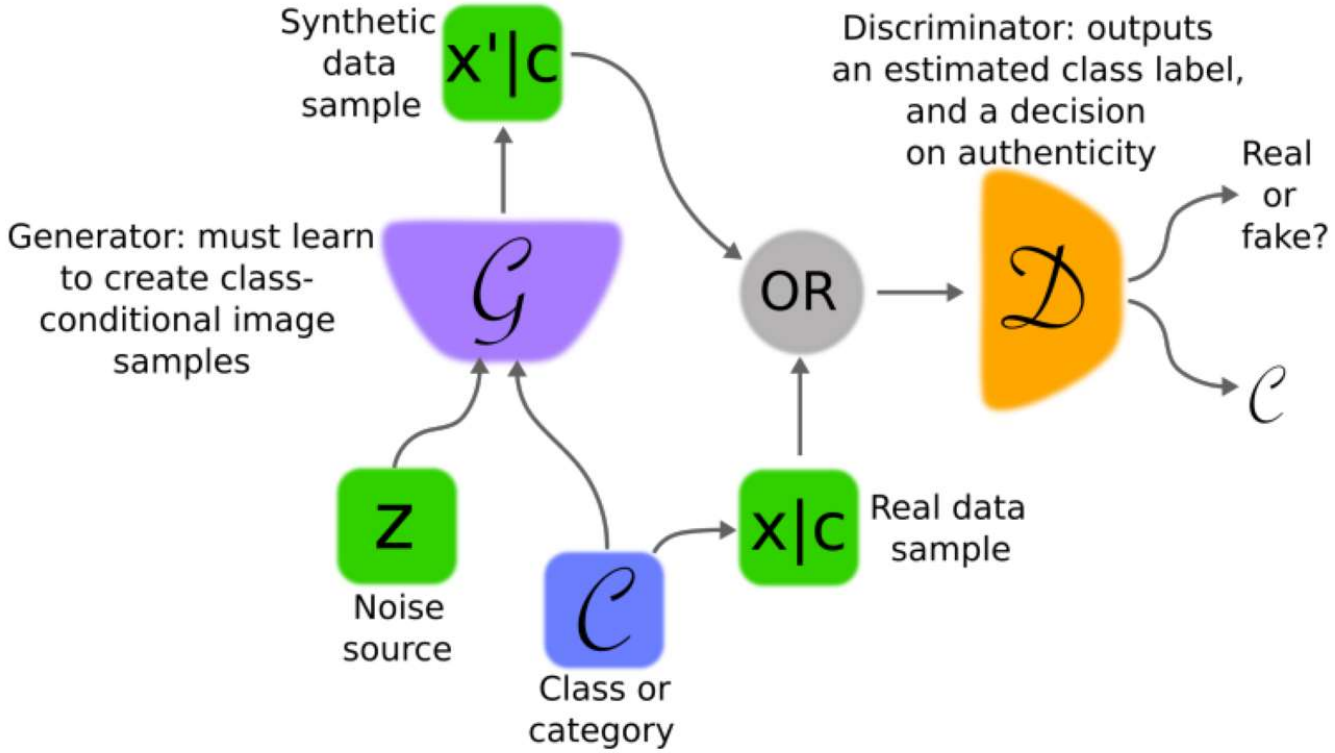
ConGAN



- data와 class label y를 concatenate하여 G와 D에 넣음.
  => G는 class label이 주어지면 해당 class의 이미지를 생성하도록 training됨.

- Discriminator는 real/fake 여부만 return함.

InfoGAN



- data와 class label y를 concatenate하여 G와 D에 넣음.
  => G는 class label이 주어지면 해당 class의 이미지를 생성하도록 training됨.

- Discriminator는 real/fake 여부와 함께 어느 class에 속하는지를 return함.

AC-GAN

$$L_s = \mathbb{E}_{p^*(x)}\left[\ln D_\psi(x)\right] + \mathbb{E}_{p_\theta(x)}\left[\ln(1 - D_\psi(x))\right]$$
$$L_c = \mathbb{E}_{p^*(x,y)}\left[\ln q_\phi(y \mid x)\right] + \mathbb{E}_{p_\theta(x,y)}\left[\ln q_\phi(y \mid x)\right]$$

- x는 image, y는 label. D는 discriminator, q는 auxilliary classifier(AC).
  p*는 real data distribution, p_theta는 fake data distribution.

- L_s는 vanilla GAN의 loss function, L_c는 'AC가 p*와 p_theta를 얼마나 잘 근사하는지'에 대한 loss function.

- Discriminator는 L_c + L_s를 maximize, Generator는 L_c - L_s를 maximize.

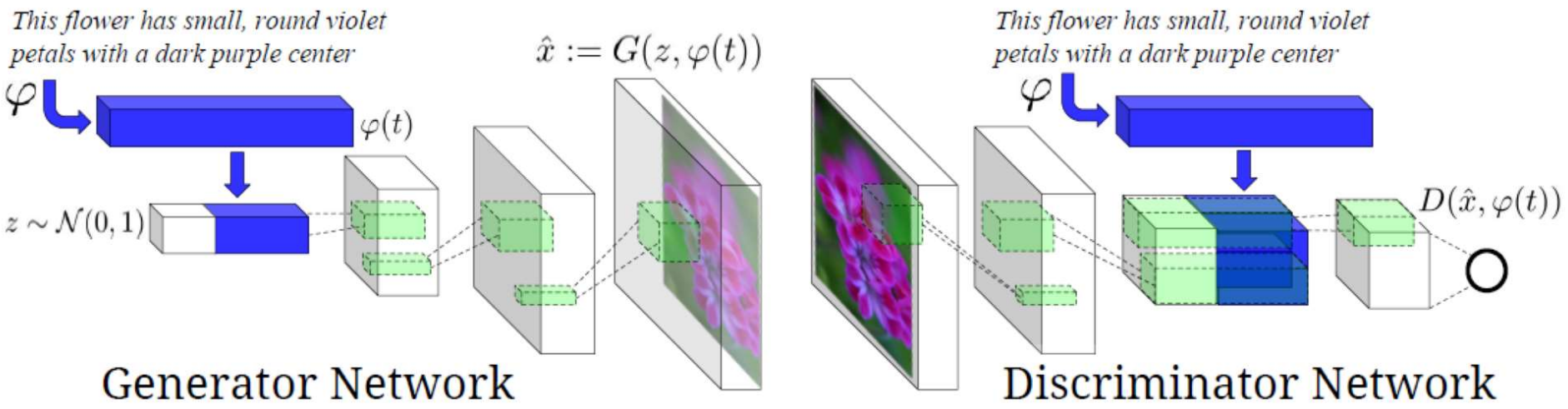
- 장점 : L_c를 maximize하는 방향으로 Generator가 training되기 때문에 class label의 boundary에서 이미지가 생성될 확률이 감소한다.
  (ex. 웃는 사람과 슬픈 표정의 사람의 경계점에서 이미지가 생성되지 않는다.)

InfoGAN vs AC-GAN

1. Real data distribution과 fake data distribution의 cross entropy를 최소화한다.
2. Classifier가 real image를 보고 label을 잘 맞춘다.
3. Classifier가 fake image를 보고 label을 잘 맞춘다.

- GAN : 1번을 목표로 G와 D를 training.
- InfoGAN : 1, 2번을 목표로 G와 D를 training.
- AC-GAN : 1, 2, 3번을 목표로 G와 D를 training.

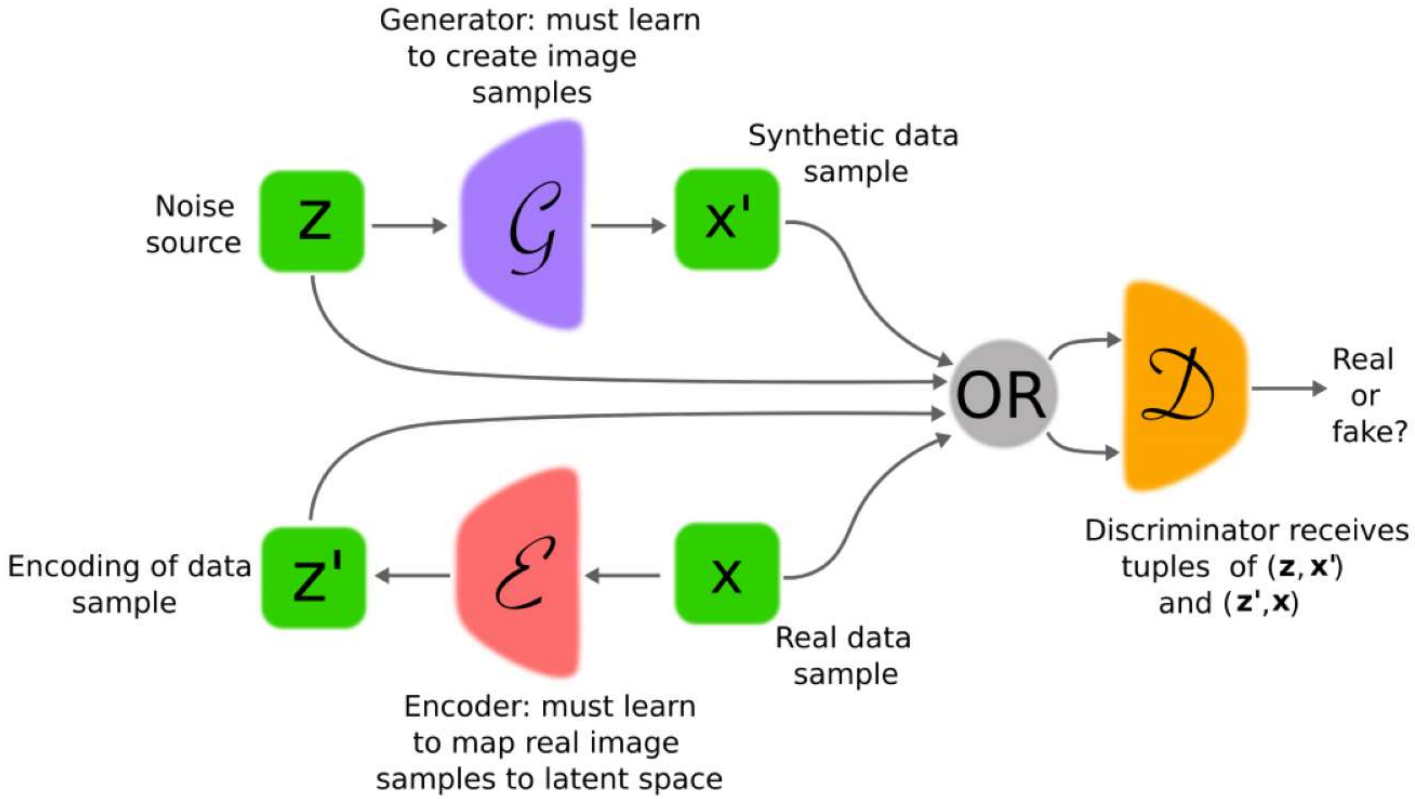
GAN-T2I



- 먼저 one-to-many를 사용하여 각 real image에 label(sentence)을 달아줌.
  그리고 image와 label의 feature를 extract.

- Generator : noise와 label의 feature를 묶어서 넣으면 fake image를 만듦.
- Discriminator : image와 label의 feature를 묶어서 넣어주면,
  해당 image가 real인지 fake인지 판별.

ALI/BiGAN



- 주어진 image가 어떤 noise sample에 대응되는지 파악하기 위해 만든 모델.
- Encoder : real image x -> noise sample z'
  Generator : noise sample z -> fake image x'
- D는 (x, z')와 (x', z)의 tuple을 받아서 둘 중 누가 real이고 누가 fake인지 구분.

WassersteinGAN

- 기존 GAN들은 두 분포 사이의 유사도를 구하기 위해 KL-divergence나 JS-divergence를 사용함. 반면, WGAN은 Wasserstein distance를 사용함.

- Wasserstein distance : $W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\| \right]$

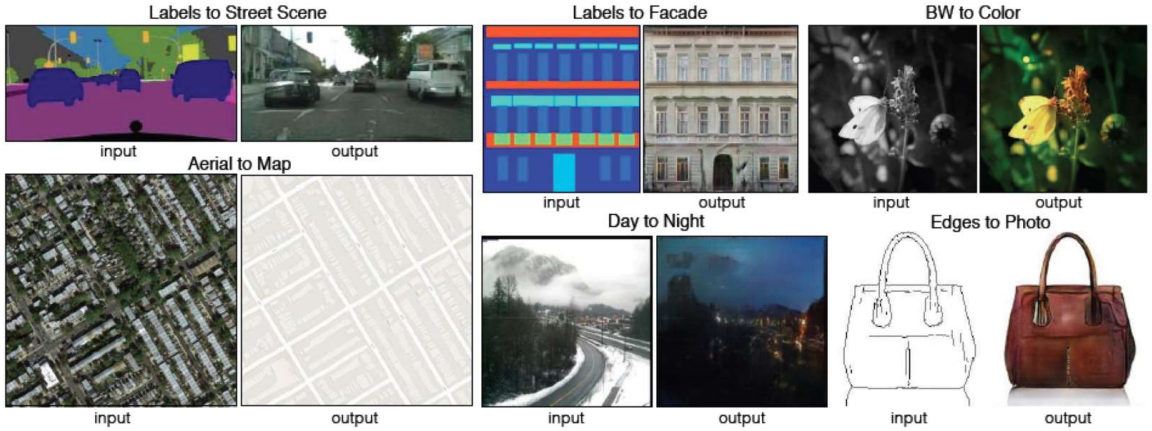
- 두 distribution이 거의 겹치지 않는 경우에 KL이나 JS는 gradient가 잘 들어오지 않음.

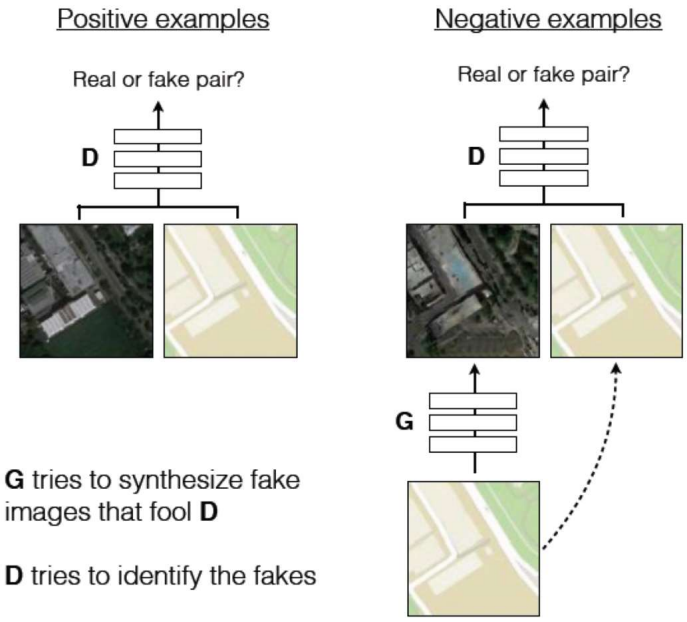
- 반면, Wasserstein은 같은 경우에도 gradient가 안정적으로 들어옴. 따라서 훨씬 많은 경우에 distribution이 수렴함. (K-Lipschitz를 만족하기 때문. 간단한 설명은 두 번째 링크 참고.)

- 장점 : 다양한 architecture에서 training이 안정적이다.

- Gradient penalty를 주는 방식으로 개선한 variant로 WGAN-GP가 있음.

Pix2pix





- Colorful Image Colorization : (흑백, 컬러) pair에서 흑백->컬러 network를 학습시킴.
  => "Image pair를 구할 수 있다면 다른 것들도 가능하지 않을까?"

- 기존 연구 : pixel-wise error의 L1 norm을 loss function으로 사용. But, not realistic.

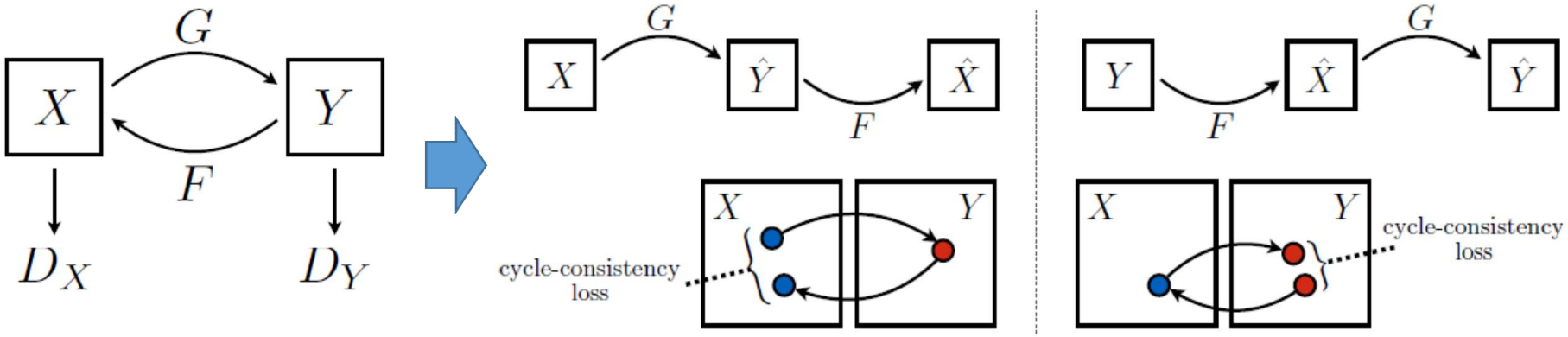
- Pix2pix : 기존의 loss function에 adversarial training을 추가함. (y : real, G(x) : fake)

$$V(G,D) = \min_G \max_D \mathbb{E}_y[log(D(y)] + \mathbb{E}_x[log(1 - D(G(x))] + \mathbb{E}_{x,y}[\|y - G(x)\|_1]$$

- Generator : U-net 사용 (Encoder와 Decoder 사이에 bypass 추가)
  Discriminator : PatchGAN 사용 (Discriminator가 image를 patch 단위로 보면서 판별)

- Training data는 약 3000개 정도.

CycleGAN



- X가 동물 사진들이고 Y가 모네의 그림들이라고 할 때, X->Y->X와 Y->X->Y 둘 다 원본과 비슷하도록 만들자는 게 Motivation! (= Cycle consistency)

- 이 방식대로라면 Pix2pix와는 달리 image pair 없이도 학습이 가능하다.

- 동물 사진을 완전히 모네의 그림으로 바꿔버리는 대신, 형태를 유지하면서 세세한 Style만 바꾸도록 유도.

- Generator(G, F)와 Discriminator(D_X, D_Y)가 각각 2개씩 존재하게 된다.

CycleGAN

$$Loss_{x \to y} = \mathbb{E}_y[log(D_y(y))] + \mathbb{E}_x[log(1 - D_y(G(x)))] + \mathbb{E}_x[\|F(G(x)) - x\|_1]$$
$$Loss_{y \to x} = \mathbb{E}_x[log(D_x(x))] + \mathbb{E}_y[log(1 - D_x(F(y)))] + \mathbb{E}_y[\|G(F(y)) - y\|_1]$$
$$Loss_{cycleGAN} = Loss_{x \to y} + Loss_{y \to x}$$

- Pix2pix의 loss function을 양방향으로 만들어서 둘의 합을 전체 loss로 정의함.
  (구현에서는 GAN loss를 Least Square로 바꿔서 안정성 높임. 맨 뒤의 링크 참고.)

- L1 norm 덕분에 원본의 전체적인 틀이 그대로 유지됨.

- Generator는 ResNet 사용, Discriminator는 PatchGAN 사용.

- Identity loss를 추가해주면 원본의 색조도 유지할 수 있음.

- 단점 : Training이 많이 느림.

DiscoGAN

(b) Handbag images (input) & **Generated** shoe images (output)

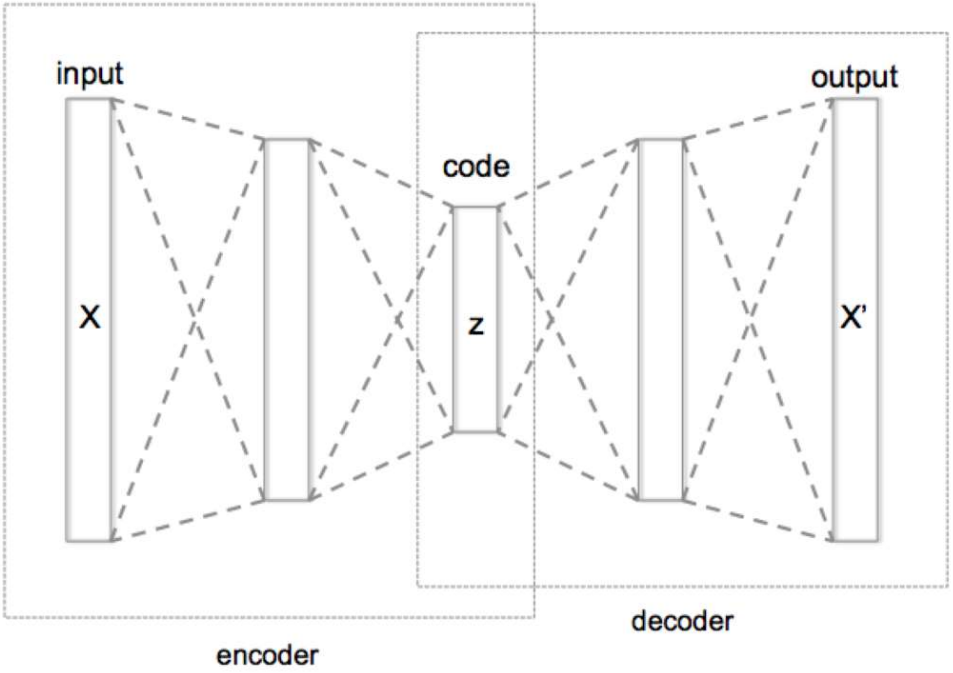
(c) Shoe images (input) & **Generated** handbag images (output)

- CycleGAN과 마찬가지로 A->B->A와 B->A->B 모두 원본과 비슷해지도록 구성 & unpaired dataset 위에서 training.

- CycleGAN과는 달리, Generator로 Encoder-decoder 구조를 쓰고 loss function 에서 L1 norm 대신 L2 norm을 사용함으로써 형태 변화가 큰 dataset에서 잘 작동하도록 구성. 반면 높은 해상도를 기대할 수는 없음.

- Network 구조가 단순해서 빠른 학습이 가능.

Auto-encoder



- 목적 : 주어진 data를 latent space의 vector로 encoding하고, latent vector를 decoding하여 원래의 data를 복구하는 것.

- encoder와 decoder는 deterministic mapping이다.

- EBGAN에서 사용함.

EBGAN

$$D(x) = ||Dec(Enc(x)) - x||. \tag{13}$$

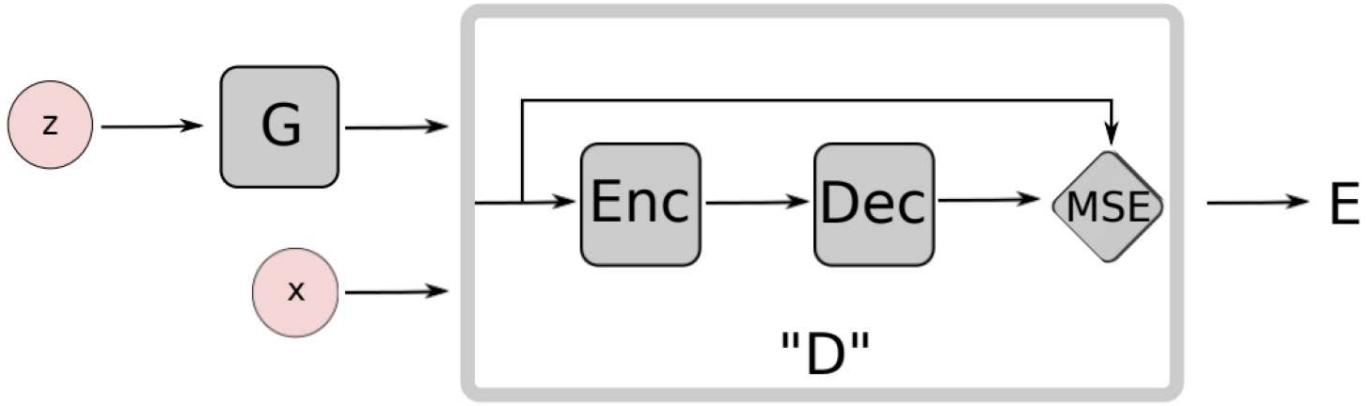


Figure 1: EBGAN architecture with an auto-encoder discriminator.

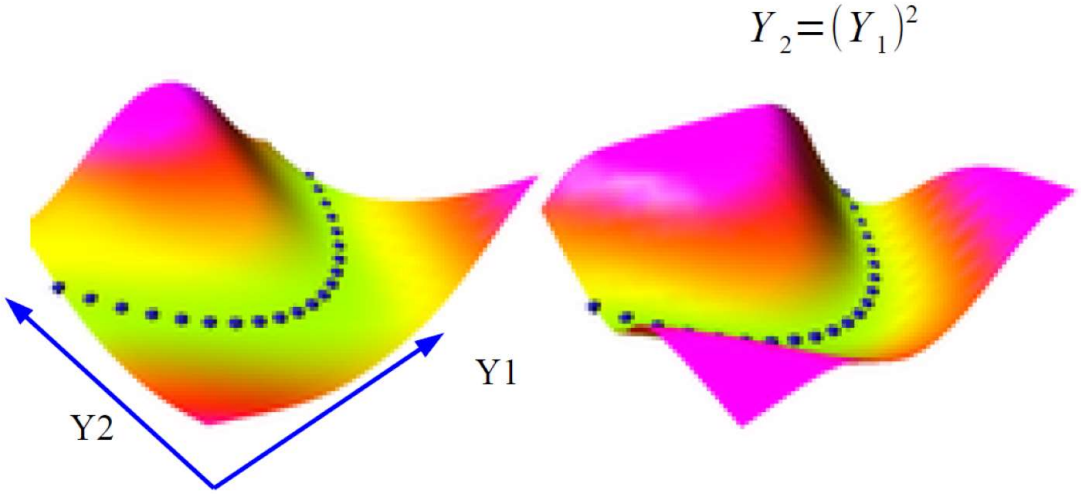
- "Energy-based"는 Discriminator가 [0,1]의 값을 갖는 대신, 주어진 image가 fake일 수록 더 큰 값(=energy)을 return한다는 데에서 온 이름.

- real image를 가지고 Discriminator를 auto-encoder로 만든다. (핵심!)

- Discriminator는 input과 Dec(Enc(input))의 mean squared error를 return한다.

- 장점 : 다양한 gradient direction이 나오며 큰 미니배치 사이즈를 사용해도 된다.
  But, 실험 결과에서 기존 연구에 비해 큰 가시적 성과는 보이지 않는 것 같음...

EBGAN



- 데이터 셋이 있을 때, 해당 데이터가 있는 지점에서는 낮은 함수값을, 그렇지 않은 지점에서는 높은 함수값을 return하는 함수를 생각해볼 수 있다.
  그게 energy function이다.

- 위 이미지에서 점을 따라서 2-dimensional data가 있을 때, 곡면이 energy function.

- real image가 주어졌을 때, auto-encoder가 저 energy function을 학습한다는 게 논문 저자들의 주장.

- V(D*, G)가 내쉬 균형일 때, G가 만든 image와 실제 image는 분포 상으로 구분할 수 없음을 증명함.

BEGAN

- 1. EBGAN처럼 Discriminator로 auto-encoder를 쓰는데,
  2. MSE를 사용하는 대신 Wasserstein distance를 사용해서
  3. real data의 분포가 아니라 auto-encoder의 loss 분포를 근사한다!

- 즉, G와 D가 training이 잘 되었다면 둘의 loss의 분포 및 기대값이 같다.

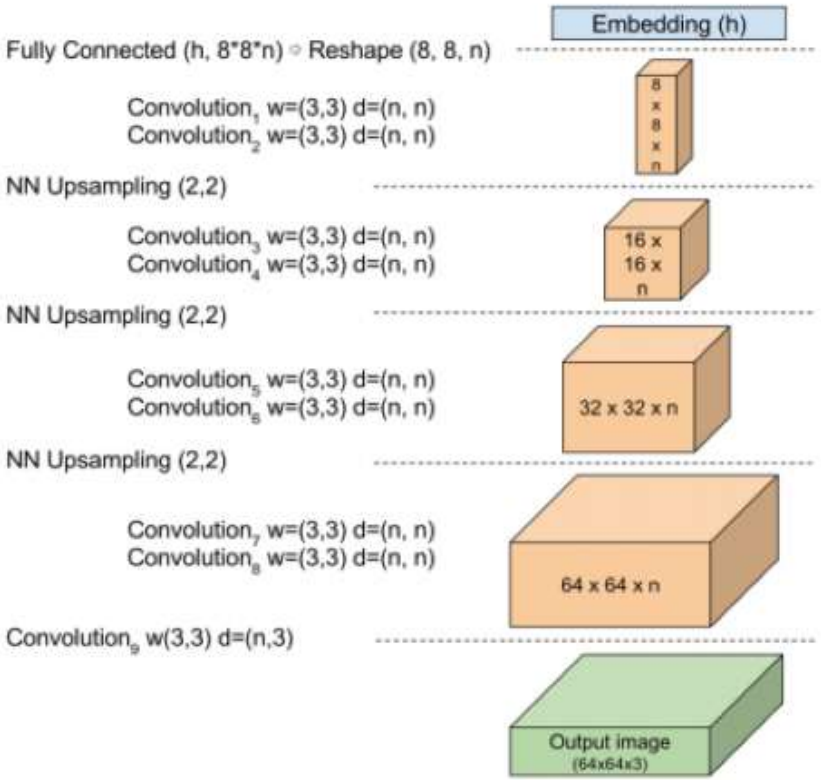
- G와 D 중 한 쪽이 training이 덜 된 상태에서 균형을 잡기 위해서 $\gamma = \frac{\mathbb{E}[\mathcal{L}(\mathcal{G}(z))]}{\mathbb{E}[\mathcal{L}(x)]}$ 이라는 parameter를 도입한다.
  gamma가 클 수록 image가 다양해지고 작을 수록 image가 realistic 해진다.

- global measure of convergence라는 식으로 모델의 수렴 정도를 알려준다.
  (= $\mathcal{M}_{global} = \mathcal{L}(x) + |\gamma\mathcal{L}(x) - \mathcal{L}(\mathcal{G}(z_{\mathcal{G}}))|$)

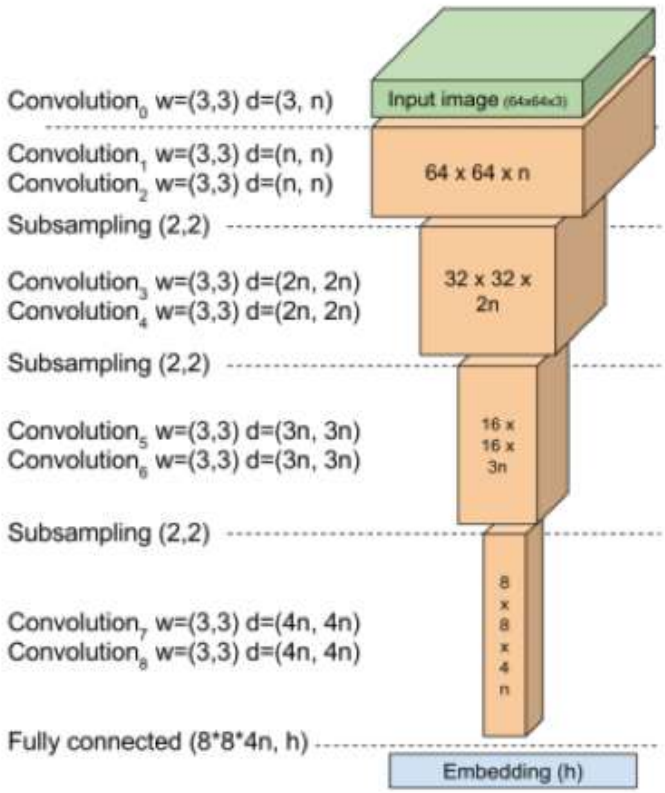
- 128x128의 고품질 이미지까지 만드는 데에 성공함.

(기호들의 정의는 위의 링크 참고)

BEGAN



(a) Generator/Decoder



(b) Encoder

(BEGAN의 구조. Batch normalization, dropout 등이 필요 없다고 함.)

출처

GAN의 기본 컨셉
- https://arxiv.org/pdf/1710.07035.pdf

DCGAN
- https://gist.github.com/shagunsodhani/aa79796c70565e3761e86d0f932a3de5

ConGAN
- https://arxiv.org/pdf/1710.07035.pdf

InfoGAN
- https://arxiv.org/pdf/1710.07035.pdf

AC-GAN
- http://ruishu.io/2017/12/26/acgan/

GAN-T2I
- https://gist.github.com/shagunsodhani/5d726334de3014defeeb701099a3b4b3

ALI/BiGAN
- https://arxiv.org/pdf/1710.07035.pdf

WassersteinGAN
- https://www.alexirpan.com/2017/02/22/wasserstein-gan.html
- https://paper.dropbox.com/doc/Wasserstein-GAN-GvU0p2V9ThzdwY3BbhoP7

출처

Pix2pix
- https://taeoh-kim.github.io/blog/gan%EC%9D%84-%EC%9D%B4%EC%9A%A9%ED%95%9C-image-to-image-translation-pix2pix-cyclegan-discogan/

CycleGAN
- https://taeoh-kim.github.io/blog/gan%EC%9D%84-%EC%9D%B4%EC%9A%A9%ED%95%9C-image-to-image-translation-pix2pix-cyclegan-discogan/

DiscoGAN
- https://taeoh-kim.github.io/blog/gan%EC%9D%84-%EC%9D%B4%EC%9A%A9%ED%95%9C-image-to-image-translation-pix2pix-cyclegan-discogan/

Auto-encoder
- https://gist.github.com/shagunsodhani/aa79796c70565e3761e86d0f932a3de5

EBGAN
- http://dogfoottech.tistory.com/183
- https://taeoh-kim.github.io/blog/generative-models-part-2-improvedganinfoganebgan/
- http://jaejunyoo.blogspot.com/search/label/GAN

BEGAN
- http://t-lab.tistory.com/27
- http://jaejunyoo.blogspot.com/2017/04/began-boundary-equilibrium-gan-2.html